Volume 10 Number 1 1982 Nucleic Acids Research

An efficient method for matching nucleic acid sequences

Joseph Felsenstein[1], Stanley Sawyer[2] and Rochelle Kochin[3]

[1]Department of Genetics, University of Washington, Seattle, WA 98195, [2]Department of Mathematics, Purdue University, West Lafayette, IN 47907, and [3]Division of Medical Genetics, University of Washington School of Medicine, Seattle, WA 98195, USA

Received 15 September 1981

ABSTRACT

A method of computing the fraction of matches between two nucleic acid sequences at all possible alignments is described. It makes use of the Fast Fourier Transform. It should be particularly efficient for very long sequences, achieving its result in a number of operations proportional to n ln n, where n is the length of the longer of the two sequences. Though the objective achieved is of limited interest, this method will complement algorithms for efficiently finding the longest matching parts of two sequences, and is faster than existing algorithms for finding matches allowing deletions and insertions. A variety of economies can be achieved by this Fast Fourier Transform technique in matching multiple sequences, looking for complementarity rather than identity, and matching the same sequences both in forward and reversed orientations.

INTRODUCTION

It will frequently be desired to find that alignment of two nucleic acid sequences which maximizes the number of base positions which are identical. The straightforward algorithm for doing this is to consider in turn each of the possible alignments, for each one running along both sequences and Mounting for each how many bases match. If there are n bases in the first sequence and m in the second, this will require nm comparisons of bases. This can be a prohibitive amount of computation: if both sequences are of length 10,000, it will require 100 million comparisons.

It might be thought that it is impossible to improve substantially on such a straightforward algorithm, but such improvement is possible. By using the Fast Fourier Transform (FFT) introduced by Cooley and Tukey (1) and described more extensively in Brigham (2), one can compute the number of matches at each possible alignment in a time proportional to n ln n. In our example involving 10,000 bases this means that the number of operations is a small multiple of 140,000, which represents a great saving.

The algorithm enables a number of economies, but there are two important tasks it cannot do. It cannot confine its interest to runs of consecutive matches: it is inherently unable to detect whether two matches are consecutive. It also cannot allow for insertions and deletions in doing the matching. Needleman and Wunsch (3) and Sellers (4, 5, 6) have presented effective methods for allowing for insertions and deletions, though their methods also require on the order of nm computational steps. Allen Delaney (personal communication) has pointed out to me that the dictionary of oligonucleotides first constructed by Korn, Queen, and Wegman (7) can be expanded and used to find the longest common subsequence of two nucleotide sequences in time proportional to $n \ln n + m \ln m$.

The present algorithm is of limited interest because of the presence of deletions and insertions in most comparisons of long nucleotide sequences, and the probability that the homology will be confined to short regions (such as coding sequences) whose presence might not be detectable if we look only at the number of matches along the entire length of a sequence. However, if the two sequences differ from each other by only a few deletions, this will be reflected in the result as substantial partial matchings at several neighboring aligments, and this would signal the presence of such deletions. Delaney's algorithm may be even more effective in detecting such cases.

THE METHOD

The procedure we follow begins by reading each sequence and constructing a series of indicators, one for each of the four bases. Each of these is an array containing a one wherever that base exists in the corresponding sequence, and a zero where it does not exist. For example, the sequence AACGUGGC has the four indicator sequences:

Sequence:	A	A	C	G	U	G	G	C
Indicator for A:	1	1	0	0	0	0	0	0
Indicator for C:	0	0	1	0	0	0	0	1
Indicator for G:	0	0	0	1	0	1	1	0
Indicator for U:	0	0	0	0	1	0	0	0

When the base is at a certain position is unknown, we have followed the practice of setting all four indicator functions to 0.25, and when we know only that the base is (say) a purine, we set two of the indicator functions to 0.5 and the others to zero. Let us denote the j-th entry in the indicator function for A as $X_j^{(A)}$, and similarly for the other four bases. The corresponding indicator function for the other sequence will be $Y_j^{(A)}$.

The number of matches of A's when the second sequence is displaced by k from the first is then given by:

$$Z_k^{(A)} = \sum_{j=1}^{n} X_j^{(A)} Y_{j+k}^{(A)} \qquad (1)$$

The overall number of matches at a shift of k is given by

$$Z_k = Z_k^{(A)} + Z_k^{(C)} + Z_k^{(G)} + Z_k^{(U)}. \qquad (2)$$

Note that the convention we have adopted for missing information implies that when an unknown base lies opposite a known one, we count one-fourth of a match.

The foundation of our method is the relation between convolutions and Fourier transforms: if X and Y are sequences whose discrete Fourier transforms are U and V, then the sequence Z giving the number of matches has the Fourier transform W, where

$$W_j = U_j V_j^*, \qquad (3)$$

where the star indicates the complex conjugate (changing the sign of the imaginary part of the complex number V_j).

Since the Fast Fourier transform algorithm allows us to compute the Fourier transform (or to invert it) in on the order of n ln n operations, this suggest the following procedure: compute the Fourier transforms of the indicator functions of the two sequences. This will be eight Fourier transforms in all. Each of these is a sequence of complex numbers. The complex conjugates of the sequences $V_j^{(A)}$, ..., $V_j^{(U)}$ are then taken (which can be done in n operations each). We could now use equation (3) to compute the Fourier transforms $W_j^{(A)}$, ..., $W_j^{(U)}$ which are the transforms of the numbers of matches of A's, C's, G's and U's at all possible shifts.

Since the Fourier transform is a linear transformation, the transform of a sum is the sum of transforms. This means that if we are interested only in the overall number of matches, without regard to which of the four nucleotides is matching, we can sum the four W's to get:

$$W_j = W_j^{(A)} + W_j^{(C)} + W_j^{(G)} + W_j^{(U)}. \qquad (4)$$

We now take the inverse Fourier transform of the sequence W_j. The real parts of the resulting sequence of complex numbers will be the numbers of matches at shifts 0, 1, ..., n. The complex parts of the result will all be zero. We have thus obtained the result we wanted with 9 Fourier transforms, each of which requires on the order of n ln n operations.

There are, however, a number of complications which must be dealt with. The FFT algorithm is most effective when the length of the sequence is a power of two. We have followed the policy of extending our indicator sequence by adding enough zeros to the end of each to make the length a power of two. Thus a sequence of 18 nucleotides leads to an indicator sequence of length 32, by adding 14 zeros at the end of each indicator array. A second problem is that when we use this technique, the resulting numbers of matches are those computed from equation (1), but with the addition of subscripts j+k being carried out modulo the length of the sequence. In effect, the sequences are taken to be circular and are rotated past one another. To avoid this, we double the length of the sequences by adding zeros.

A sequence of 28 bases then requires four indicator sequences, each of length 64. If there are two sequences of length 28, the resulting sequence of numbers of matches will have (in this case) the matches for rightwards shifts of the second sequence by amounts 0, 1, 2, ..., 27, followed by 8 zeros, and then the values for shifts of -27, -26, ..., -2, -1.

There are a number of inefficiencies involved. All the arithmetic uses real numbers rather than integers, and these computations are slower and the numbers take up as much as twice the memory space as integers. We must compute and store nine complex sequences, so that the minimum storage requirement is 18n real numbers. However, the intrinsic advantage of the algorithm is that the number of operations grows as n ln n, and it must ultimately be faster than an algorithm which requries of the order of n^2 operations, provided that we take a large enough n.

COMPUTATIONAL EXPERIENCE

A program has been written in Pascal by R. K., and its performance compared to the performance of a program written to carry out the simple algorithm. Both programs have been run on the same (artificial) data set on a DEC VAX 11/780. The results in CPU seconds are, for two sequences each of length n:

n	FFT Method	Simple Method
100	2.99	0.54
1000	18.48	11.64
2000	40.16	43.77
4000	90.57	169.58

Once we have more than 1000 nucleotides, both algorithms show their predicted dependence on n, the FFT method rising a bit faster than linearly and the simple matching method rising as n^2. Since it involves real rather than integer arithmetic, the FFT method is slower for small data sets. The crossover point is just below 2000 nucleotides.

A listing of the Pascal program may be obtained by writing to R. K. or to J. F.

ADDITIONAL OBJECTIVES

One of the advantages of the FFT approach is that a variety of additional objectives can be accomplished without much additonal computational effort. Once the Fourier transforms of the indicator arrays have been computed, these can be re-used in a number of ways:

(1) If we have multiple sequences, and wish to look at all possible pairs of sequences, we can proceed from the transforms of each sequence without the need to recompute these. If there are s sequences, we will have to do s(s-1)/2 back Fourier transforms, but only s forward Fourier transforms.

(2) If we wish to match two sequences with one of them in reverse orientation, the procedure for doing this differs only at the step corresponding to equation (3), which becomes

$$W_j = U_j V_j, \tag{5}$$

the complex conjugate of V not being taken. Thus once we have obtained the Fourier transforms U and V, these can be used to do matching in both orientations. Thus while it requires 9 Fourier transforms to do matching in one orientation, it requires only 10 to do it in both orientations.

(3) If we wish to look not only for identical bases, but for complementary bases as well, we can again re-use the forward transforms. The number of complementary bases at shift k is given by

$$Z_k' = \sum_{j=1}^{n} [X_j^{(A)} Y_{j+k}^{(U)} + X_j^{(C)} Y_{j+k}^{(G)} + X_j^{(G)} Y_{j+k}^{(C)} + X_j^{(U)} Y_{j+k}^{(A)}] \tag{6}$$

This can be computed by, at the step corresponding to equation (3), forming the product, not of indicator sequences of identical bases, but of complementary bases. Thus we again do not need to recompute the forward Fourier transforms, but only need to perform the appropriate product and do a back transform.

If we want to look at both identity and complementarity, and at both forward and reverse orientations, we need only 12 Fourier transforms in all.

It may be worth noting that we can use the products of transformed sequences directly, without back transformation, to test for the statistical significance of the amount of matching or complementarity. Such a test would utilize standard time series statistical methods, under the assumption that the indicator sequences could be considered as normally distributed to a sufficient degree of approximation.

LIMITATIONS

The primary limitation of this method is that what it computes is of limited value. In the evolution of nucleic acid sequences, deletion and insertion play a large role, and we do not expect to find that two sequences differ only by a shift plus some base changes. If there are only a few deletions expected, then it is likely that the two sequences are relatively short, and the present methods have little advantage. It is unfortunate that so many of the algorithms which are likely to apply to real data require computational effort on the order of n^2 or n^3.

The present method may be of some value in exploring sequences and looking for alignments which have a significant amount of matching or complementarity. If there has been only one deletion, we would expect to find two nearby alignments, both of which show significant matching or significant complementarity. As more and more deletions separate two sequences, the ability to use this technique for exploratory purposes would degrade rapidly.

ACKNOWLEDGMENTS

We wish to thank Charles Langley, Walter Fitch, Allen Delaney, and Temple Smith for helpful, if not entirely encouraging, comments. Work by J. F. was supported by task agreement number DE-AT06-76EV71005 of contract number DE-AM06-76RL02225 between the U. S. Department of Energy and the University of Washington. S. S. was supported by NSF grant MCS81-03472, and R. K. by U. S. Public Health Service grant number GM 15253, principal investigator Arno Motulsky.

REFERENCES

1. Cooley, J.W. and Tukey, J.W. (1965) Math. Comp. 19, 297-301.
2. Brigham, E.O. (1974) The Fast Fourier Transform. Prentice-Hall, Englewood Cliffs, New Jersey.
3. Needleman, S.B. and Wunsch, C.D. (1970) J. Mol. Biol. 48, 443-453.
4. Sellers, P.H. (1974) J. Combinator. Theor. 16, 253-258.
5. Sellers, P.H. (1974) SIAM J. on Appl. Math. 26, 787-793.
6. Sellers, P.H. (1979) Proc. Natl. Acad. Sci. USA 76, 3041.
7. Korn, L.J., Queen, C.L., and Wegmann, M.N. (1977) Proc. Natl. Acad. Sci. USA 74, 4401-4405.